Questão 3 -

X ← variável aleatória que representa a execução da linha 4

$X_i = \begin{cases} 1 & \text{se a linha 4 executar pelo menos } i \text{ vezes} \\ 0 & \text{cc} \end{cases}$

$X_i = \Pr\{X_i = 1\} = \frac{1}{i}$ = Probabilidade de $v[i]$ ser máximo em $v[1..i]$

$E[X] = \sum_{k=1}^{n} X_i = \sum_{i=1}^{n} \Pr\{X_i = 1\} = \sum_{i=1}^{n} \frac{1}{i} < 1 + \lg n$

$E[X] = O(\lg n)$

Questão 4 -

X - variável aleatória que representa a execução das linhas 5 e 8

$X_i \begin{cases} 1, & \text{se as linhas 5 e 8 executam pelo menos } i \text{ vezes} \\ 0 & \text{ec} \end{cases}$

$X_i = \Pr\{X_i = 1\}$ = Probabilidade da linha 5 executar + Probabilidade da linha 8 executar

Probabilidade da linha 5 executar = Probabilidade de $v[i]$ ser máximo em $v[1..i]$

Probabilidade linha 5 $= \frac{1}{i}$

Probabilidade linha 8 $= \frac{1}{i}$, já que $v[i]$ não é o maior

$\Pr\{X_i = 1\} = \frac{1}{i} + \frac{1}{i} = \frac{2}{i}$

$E[X] = \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} \frac{2}{i} = 2 \cdot \sum_{i=1}^{n} \frac{1}{i} \leq 2 \cdot (\lg n + 1)$

$E[X] = O(\lg n)$

Questão 5 -

X ← variável aleatória que representa nº de execuções da linha 6.

$X_i = \begin{cases} 1 & \text{se a linha 6 é executada pelo menos } i \text{ vezes} \\ 0 \end{cases}$

$X_i = \Pr\{X_i = 1\} = \frac{1}{i}$

$E[X] = \sum_{i=1}^{n} X_i = \frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n} \leq \lg n + 1$ //

$E[X] = O(\lg n)$

Y ← variável aleatória que representa nº de execuções da linha 7

$Y_i = \begin{cases} 1 & \text{se a linha 7 é executada} \\ 0 & \text{cc} \end{cases}$

$Y_i = \Pr\{Y_i = 1\} = \frac{1}{i}$

$E[Y] = \sum_{i=1}^{n} Y_i = \frac{1}{1} + \frac{1}{2} + \frac{1}{n} \leq \lg n + 1$

$E[Y] = O(\lg n)$

PROVA 2002/1

Questão 1 -

```
ORDENA(A, n)
1 j ← n-1  min ← n
2 enquanto j >= 0
3   K ← BUSCAMIN(A, 0, j)
4   se K = 0
5     então FLIP(j, A, n)
6     senão se K > 0 e K < j
7       FLIP(K, A, n)
8       FLIP(j, A, n)
9   j ← j - 1
```

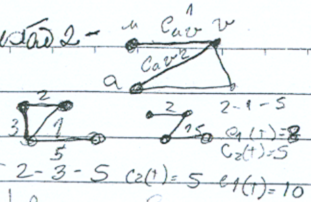
tamanho da instância: n

$T(n)$ = nº de chamadas de FLIP

linha	$T(n)$
1	$O(1)$
2-4	$O(n)$
5	$O(n)$
7-8	$2O(n)$
9	$O(n)$

consumo total $= T(n) = 2O(n) = O(n)$

Questão 2 -



$c_1(T) := \Sigma \{ c_{uv} : uv \in T$

$c_1(T) := c_{uv} + c_{av}$

$c_2(T) := \max\{ c_{uv} : uv \in T$

$c_2(T) := c_{av} = 2$

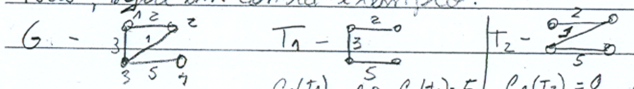
Problema 1: Encontrar uma árvore geradora T de G que minimize $c_1(T)$.

Problema 2: Encontrar uma árvore geradora T de G que minimize $c_2(T)$.

Resposta:

→ Sim, pois se encontrarmos uma árvore geradora mínima quer dizer que teremos arestas com os menores custos possíveis minimizando consequentemente o maior custo de uma aresta em T.

→ Não, segue um contra-exemplo:



$c_1(T_1) = 10$ $c_2(T_1) = 5$ | $c_1(T_2) = 8$ $c_2(T_2) = 5$

Neste caso não importa se a árvore é mínima a aresta de maior custo é justamente uma aresta insubstituível.

→ O algoritmo de Kruskal que consome tempo $O(E \lg V)$ onde E é o conjunto de arestas e V o nº de vértices considerando que $|E| < |V|^2$

Questão 3 -

1- Verdadeiro. Se B tem uma solução polinomial então como A pode ser reduzido a B consequentemente A também terá uma solução polinomial.

2- Falso se $P \neq NP$. Se A pode ser reduzido a B então A é "mais difícil" que B. Portanto, se B está em NP, A não pode estar em P.

3- Falso. Se B é NP-completo então B está em NP e qualquer problema NP pode ser reduzido a B então podemos afirmar é que A está em NP.

4. Verdadeiro. Há problemas em NP que não podem reduzir polinomialmente um outro problema NP

5- Falso se $P \neq NP$. Se existirem problemas NP-completos em P, então todo problema em NP teria solução polinomial ou seja, $P = NP$.

Dúvida - Perguntar Talita → quantidade de números.

Questão 4 - determina (A, n, j)

```
1     min ← +∞
2     Para i ← 1 até n-j+1
3         Soma ← 0
4         Para v ← i até j+i-1
5             Soma ← Soma + A[v]
6         Se min > Soma
7             T ← i v j+i-1
```

~~Tamanho da~~ Instância : n

$T(n)$ = Consumo de Tempo

Linha	$T(n)$
1	$O(1)$
2-3	$n-j+1$
4	$j \cdot (n-j+1)$
6-7	$(n-j+1)$

Caso $j = 1$ $T(n) = (n-j+1) \cdot (j) = nj - j^2 + j = n$ /

Caso $j = n$ $T(n) = (n-j+1)(j) = n^2 - n^2 + n = n$ //

$j = n/2$ $T(n) = n \cdot j - j^2 + j = n \cdot \frac{n}{2} - \left(\frac{n}{2}\right)^2 + \frac{n}{2} = \frac{n^2}{2} - \frac{n^2}{4} + \frac{n}{2} = \frac{2n^2 - n^2}{4} + \frac{n}{2} = \frac{n^2}{4} + \frac{n}{2}$

Questão 5-

Tamanho da instância: $n = fim - ini + 1$

Seja $T(n)$ o consumo de mergesort-no-lugar no pior caso

$T(n) = 0$ para $n = 1$

$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta n^2$ para $n \geq 2^1, 2^2 \ldots$

Para simplificar:

$T(n) = 0$ para $n = 1$

$T(n) = 2T(n/2) + n^2$ para $n \geq 2^1, 2^2 \ldots$

$= 2^2 T(n/4) + 2 \cdot (n/2)^2 + n^2$

$= 2^3 T(n/8) + 2^2 (n/4)^2 + 2 \cdot (n/2)^2 + n^2$

$= 2^i T(n/2^i) + \frac{1}{4} n^2 + \frac{1}{2} n^2 + n^2$

$= 2^i T(n/2^i) + n^2 \cdot \sum_{k=0}^{i-1} \frac{1}{2^k}$

$= 2^{\lg n} T(1) + n^2 \cdot \sum_{k=0}^{\lg n - 1} \left(\frac{1}{2}\right)^k$

$= 0 + n^2 \cdot \left(\left(\frac{1}{2}\right)^{\lg n} - 1\right) \cdot (-2)$

$= n^2 \cdot \left(\frac{1}{n} - 1\right) \cdot (-2)$

$= -2n + 2n^2$

Vou provar que a fórmula fechada está correta.

Prova por indução em n:

base: $n = 1$

$T(n) = 0 = -2 \cdot 1 + 2 \cdot 1^2 = 0$ ok

passo: $n \geq 2^1, 2^2 \ldots$

$T(n) = 2T(n/2) + n^2$

$\stackrel{H.I}{=} 2 \cdot \left(-2 \cdot n/2 + 2 \cdot (n/2)^2\right) + n^2$

$= 2 \cdot (-n + n^2/2) + n^2$

$= -2n + 2n^2/2 + n^2$

$= -2n + n^2 + n^2$

$= -2n + 2n^2$ //

É fácil concluir que $T(n) = O(n^2)$

Questão 6-

Algoritmo F1

Para uma instância (x, y) Seja $T(x, y)$ = consumo de tempo de F1 no pior caso.

$T(x, y) = 0$ se $x = 1$ ou $y = 1$ $\quad 2^3 \quad 8 \quad \max\{x, y\} + 1$

$T(x, y) = T(x-1, y) + T(x, y-1) + 1$

$= T(x-2, y) + T(x-1, y-1) + T(x-1, y-1) + T(x, y-2) + 1 + 1 + 1$

$x - 1 = 0$

$x = i$

$= T(x-3, y) + T(x-2, y-1) + 2[T(x-2, y-1) + T(x-1, y-2) + 1] + T(x-1, y-2) + T(x, y-3)$

$+ 1 + 1 + 1 + 1 + 1$

$= 0 + 3[T(x-3, y-1) + T(x-2, y-2) + 2 + 2] + T(x-2, y-2) + T(x-1, y-3) + T(x-2, y-2)\ T(x-1, y-3)$

$> 2^{\min\{x, y\}}$

$= \Omega(2^{\min(x, y)})$ = exponencial

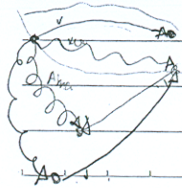
F2

Para instância (x, y). Seja $T(x, y)$ - consumo de tempo de F2 no pior caso.

Linha	$T(x, y)$
1	$O(x)$
2	$O(y)$
3	$O(x)$
4-5	$O(x \cdot y)$
Total	$T(x, y) = O(x \cdot y)$

O F2 é mais eficiente pois utiliza programação dinâmica consumindo no máximo $(x \cdot y)$ de tempo já F1 tem consumo exponencial calculando várias funções repetidas vezes.

Questão 8- Algoritmo (G, V, E, u, v)



1. Verifica todos os arcos adjacentes a u e v

se não existir arco amarelo

devolver corte vermelho-verde

se não existir arco vermelho

devolve "circuito verde-amarelo"

2 Para cada vértice K em Adj[u]

aresta ← (u,K)

Enquanto aresta ≠ NULL ou aresta ≠ uv

Se aresta = vermelho

vermelho ← 1 amarelo ← 0

senão se aresta = amarelo

amarelo ← 1 vermelho ← 0

se vermelho = 1 e amarelo = 0 e aresta = NULL

devolve vermelho-verde

se vermelho = 0 e amarelo = 1 e aresta = uv

devolve circuito verde-amarelo

aresta ← lpos(aresta)

Consumo

$O(|E|) + O(|E| \cdot |E|)$

Questão 8 - Pg. 64

Prova 2001/2

Questão 1 - O limitante inferior para uma ordenação baseada em comparação é $\Omega(n \lg n)$, baseado nisso temos como afirmativa $O(n \lg \sqrt{n})$ como um suposto limite para um algoritmo de ordenação.

O professor Progresso está certo se $n \lg \sqrt{n} = \Omega(n \lg n)$

Vamos provar que $n \lg \sqrt{n} \geq \frac{1}{4} n \lg n$ para $n \geq 1$

$n \lg \sqrt{n} = n \lg n^{1/2}$

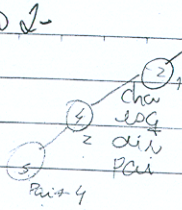
$= n \frac{1}{2} \lg n$

$= \frac{1}{2} n \lg n$

$\geq \frac{1}{4} n \lg n$

Então o professor está certo.

Questão 2 -



Algoritmo(x, y)

1 se x ≠ NULL

2 Algoritmo(esq(x), x)

3 Algoritmo(dir(x), x)

4 pai(x) ← y

O algoritmo recebe uma raiz e um pai = NULL, e devolve a árvore com o campo pai preenchido.

Consumo de tempo. Tamanho instância: n

Linha	T(n)
1	$O(1)$
2	$T(K)$
3	$T(n-K-1)$
4	$O(1)$

Teorema: Se x é raiz de subárvore de n nós então Algoritmo consome $O(n)$.

Prova: Seja x a raiz de uma árvore vazia então o consumo é zero

$T(n) = 0$

Seja a raiz diferente de NULL, a subárvore esquerda com K nós e a subárvore direita com $(n-K-1)$ nós então $T(n) = T(K) + T(n-K-1) + \Theta(1)$.

Vamos provar que $T(n) \leq c_1 . n$ para $n \geq 0, c_1 \geq 1$

$T(n) = T(K) + T(n-K-1) + 1$

$\leq c_1 . K + c_1 .(n-K-1) + 1$

$= c_1 K + c_1 . n - c_1 K - c_1 + 1$

$= c_1 . n - c_1 + 1$

$\leq c_1 . n$ //

$n^{n/2} \cdot n^{n/2} = n^{\frac{n}{2}+\frac{n}{2}}$

$\underbrace{n.n \mid n.n}_{n}$

Questão 3 - EXPOENTE(n, n0)

se n = 0

então devolva 1

se n = 1

então devolva n0

se n > 1

então exp ← EXPOENTE(⌊n/2⌋, n0).

se (n/2 % ⌊n/2⌋) = 0

então exp ← exp * exp

senão exp ← exp * exp * n0

devolve exp

3^3

$\frac{3}{2}$ 1

1,5 ≠ 1

3x3x3